제 12 장
표현 언어(EL)

2020년도 1학기

무슨 필요?

```
String name = request.getParameter("name");
String studentNum =
                    request.getParameter("studentNum");
String sex = request.getParameter("sex");
String country = request.getParameter("country");
```

- 성명 : <%= name%><p>
- 학번 : <%= studentNum%><p>
- 성별 : <%= sex%><p>
- 국적 : <%= country%><p>

편의성 증진 효과

8줄짜리 코드가
4줄로!

- 성명 : ${param.name}<p>
- 학번 : ${param.studentNum }<p>
- 성별 : ${param.sex}<p>
- 국적 : ${param.country}<p>

속성값 읽거나 쓰는 경우

scope의 네 가지 종류별로 각각 별도로 처리

A NOTE ABOUT SCOPES

There are four different attribute scopes mentioned in this section (page, request, session, and application), but you may not understand the difference between them or what exactly they are. Each of these scopes has a progressively larger and longer scope than the previous one. You should already be familiar with the request scope: It begins when the server receives the request and ends when the server completes sending the response back to the client. The request scope exists anywhere that has access to the request object, and the attributes bound to the request are no longer bound after the request completes.

In Chapter 5 you learned about sessions and session attributes, so by now you may have figured out that the session scope persists between requests, and that any code with access to the `HttpSession` object can access the session scope. When the session has been invalidated, its attributes are unbound and the scope ends.

The page and application scopes are somewhat different. The page scope encapsulates attributes for a particular page (JSP) and request. When a variable is bound to the page scope, it is available only to that JSP page and only during the life of the request. Other JSPs and Servlets cannot access the page scope-bound variable, and when the request completes, the variable is unbound. With access to the `JspContext` or `PageContext` object, you can store and retrieve attributes that exist within the page scope using the `setAttribute` and `getAttribute` methods. The application scope is the broadest scope, existing across all requests, sessions, JSP pages, and Servlets. The `ServletContext` object you learned about in Chapter 3 represents the application scope, and attributes that are stored in it live in the application scope.

여러 번의 접속이 필요한 웹페이지의 경우, 맥주 추천사이트

EL을 위한 내장객체

- pageScope
- requestScope
- sessionScope
- applicationScope

- param
- paramValues

- header
- headerValues
- cookie
- initParam

- pageContext

표현 언어란?

- JSP에서 브라우저의 출력은 주로 표현식 태그를 이용
  - <%= request.getParameter(“userid”) %>
- 표현언어(Expression Language)를 이용
  - ${ param.userid }
  - ${ param[‘userid’] }
  - ${ param[“userid”] }
- 표현언어
  - <%= %>인 표현식 대신에 사용하거나
  - 내장객체 또는 액션태그에 저장된 자료를 쉽게 참조하기 위해 만들어진 언어

표현 언어 문법 (규칙)

- ${ *exp* }
  - 표현언어는 $로 시작
  - 표현언어의 문장구조는 ${ *exp* }
  - 표현식 exp에서는 산술, 관계, 논리와 같은 기본적인 연산이 가능
- 자료유형
  - 정수형
  - 실수형
  - 문자열형
  - true, false의 논리(Boolean)형
  - null 값
- 상수
  - 논리값(boolean) true, false
  - 자바에서 이용되는 정수형으로 1, -5
  - 자바에서 이용되는 실수형으로 3.1, 4.5E+4
  - 문자열은 ‘java’, “java”와 같이 큰 따옴표, 작은 따옴표 모두 이용 가능
  - 아무것도 없다는 의미의 null

표현 언어 연산자

다양한 연산자

반환 유형	메소드 이름
이항 산술 연산자	+ - * / div % mod
이항 관계 연산자	< <= == != >= > lt le eq ne ge gt
첨자 연산자	. []
이항 논리 연산자	&& and || or
단항 논리 연산자	! not
단항 산술 연산자	-
empty 연산자	empty
삼항 조건 연산자	? :
괄호 연산자	()

우선순위 존재함 (교재 390쪽 표 12-2)

표현언어 Basic

- 기본 연산식 예시
- **392쪽 예제 12-1**
  - basicEL.jsp

- 피연산자가 비어있는지 확인
- **395쪽 예제 12-2**
  - emptyEL.jsp

표현 언어 내장 객체

분류	내장객체	자료유형	기능
JSP page 객체	pageContext	javax.servlet.jsp.PageContext	JSP 페이지 기본 객체로서, servletContext, session, request, response 등의 여러 객체를 참조 가능
범위	pageScope	java.util.Map	page 기본 객체에 저장된 속성의 <속성, 값>을 저장한 Map 객체, ${pageScope.속성}으로 값을 참조
	requestScope	java.util.Map	request 기본 객체에 저장된 속성의 <속성, 값>을 저장한 Map 객체, ${pageScope. 속성}으로 값을 참조
	sessionScope	java.util.Map	session 기본 객체에 저장된 속성의 <속성, 값>을 저장한 Map 객체, ${sesssionScope. 속성}으로 값을 참조
	applicationScope	java.util.Map	Application 기본 객체에 저장된 속성의 <속성, 값>을 저장한 Map 객체, ${applicaionScope. 속성}으로 값을 참조
요청 매개변수	param	java.util.Map	요청 매개변수 <매개변수이름, 값>을 저장한 Map 객체, ${param.name}은 request.getParameter(name)을 대체
	paramValues	java.util.Map	요청 매개변수 배열을 <매개변수이름, 값>을 저장한 Map 객체, request.getParameterValues() 처리와 동일
요청 헤더	header	java.util.Map	요청 정보의 <헤더이름, 값>을 저장한 Map 객체, ${header["name"]}은 request.getHeader(헤더이름)와 같음
	headerValues	java.util.Map	요청 정보 배열을 <헤더이름, 값>을 저장한 Map 객체, request.getHeaders()의 처리와 동일
초기화 매개변수	initParam	java.util.Map	초기화 매개변수의 <이름, 값>을 저장한 Map 객체, ${initParam.name}은 application.getInitParameter(name)을 대체
쿠키	cookie	java.util.Map	쿠키 정보의 배열을 <쿠키이름, 값>을 저장한 Map 객체, request.getCookies()의 Cookie 배열의 이름과 값으로 Map을 생성

표현식 구조 고찰

${앞단어.뒷단어}

내장 객체
- pageScope
- requestScope
- sessionScope
- applicationScope
- param
- paramValues
- header
- headerValues
- cookie
- initParam
- pageContext

속성
- Page Scope 안에서
- Request Scope 안에서
- Session Scope 안에서
- Application Scope 안에서

Map<K,V>

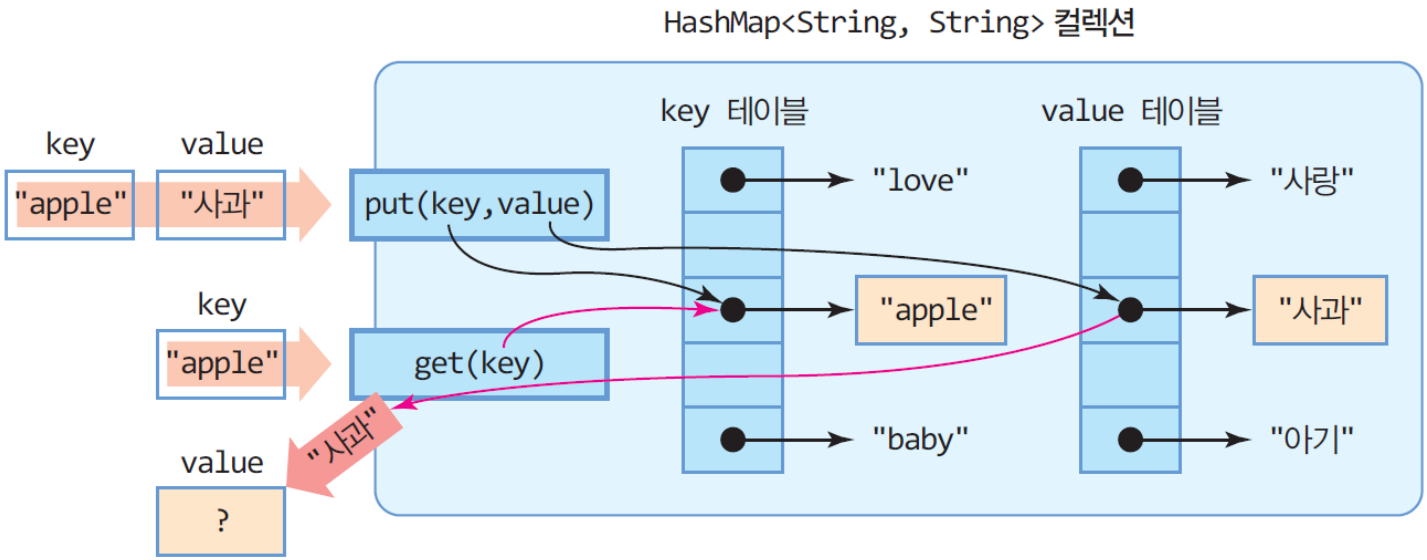
Map<K,V>

- 키(key)와 값(value)의 쌍으로 구성되는 요소를 다루는 컬렉션
  - java.util.Map
  - K는 키로 사용할 요소의 타입, V는 값으로 사용할 요소의 타입 지정
  - 키와 값이 한 쌍으로 삽입
  - 키는 해시맵에 삽입되는 위치 결정에 사용
  - 값을 검색하기 위해서는 반드시 키 이용
- 삽입, 삭제, 검색이 빠른 특징
  - 요소 삽입 : put() 메소드
  - 요소 검색 : get() 메소드
- 예) Map<String, String> 생성, 요소 삽입, 요소 검색

```
HashMap<String, String> h = new HashMap<String, String>();
h.put("apple", "사과"); // "apple" 키와 "사과" 값의 쌍을 해시맵에 삽입
String kor = h.get("apple"); // "apple" 키로 값 검색. kor는 "사과“
```

Map<String, String>의 내부 구성

Map<String, String> map = new Map<String, String>();



Map<K,V>의 주요 메소드

메소드	설명
void clear()	해시맵의 모든 요소 삭제
boolean containsKey(Object key)	지정된 키(key)를 포함하고 있으면 true 리턴
boolean containsValue(Object value)	지정된 값(value)에 일치하는 키가 있으면 true 리턴
V get(Object key)	지정된 키(key)의 값 리턴, 키가 없으면 null 리턴
boolean isEmpty()	해시맵이 비어 있으면 true 리턴
Set<K> keySet()	해시맵의 모든 키를 담은 Set<K> 컬렉션 리턴
V put(K key, V value)	key와 value 쌍을 해시맵에 저장
V remove(Object key)	지정된 키(key)를 찾아 키와 값 모두 삭제
int size()	HashMap에 포함된 요소의 개수 리턴

표현언어 내장객체 이용

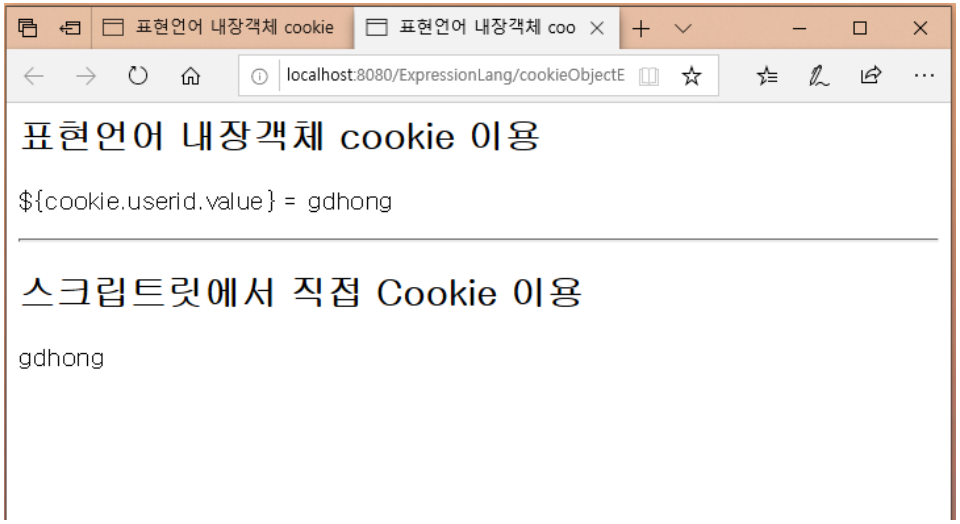
- requestScope
  - 400쪽 예제 12-3
  - implicitObjectEL.jsp

EL Implicit Object	Result
${empty param.age}	true
${!empty param.age}	false
${pageContext.request.contextPath}	/ExpressionLang
${requestScope.univ}	한국대학교
${requestScope['univ']}	한국대학교
${applicationScope.univ}	동양미래대
${applicationScope.name}	홍길동

표현언어 내장객체 이용

cookie

- cookieObjectEL.jsp
- 402쪽 예제 12-4

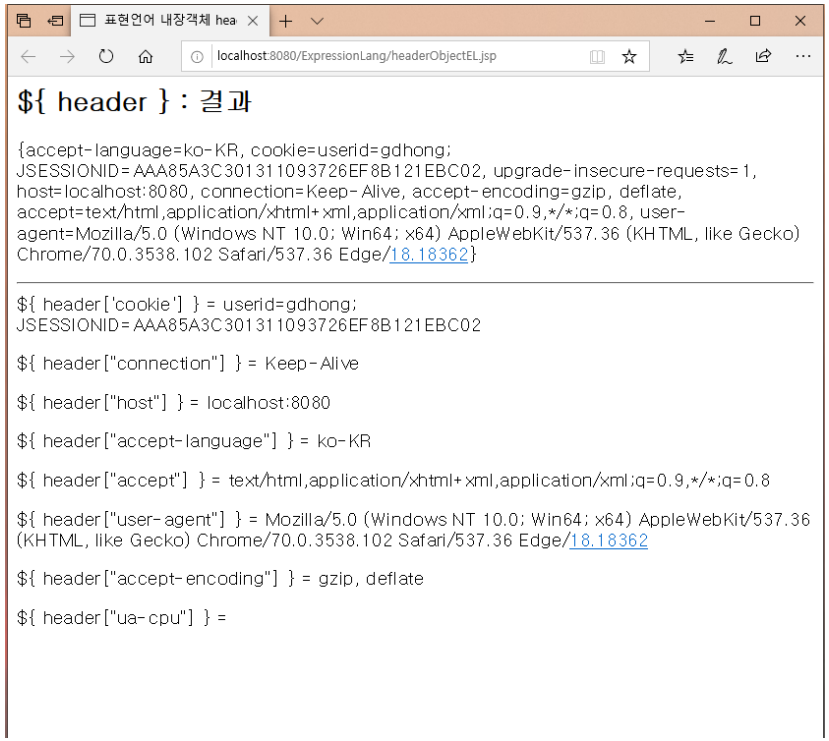


코드 상의 장점

표현언어 내장객체 이용

header와 headerValues

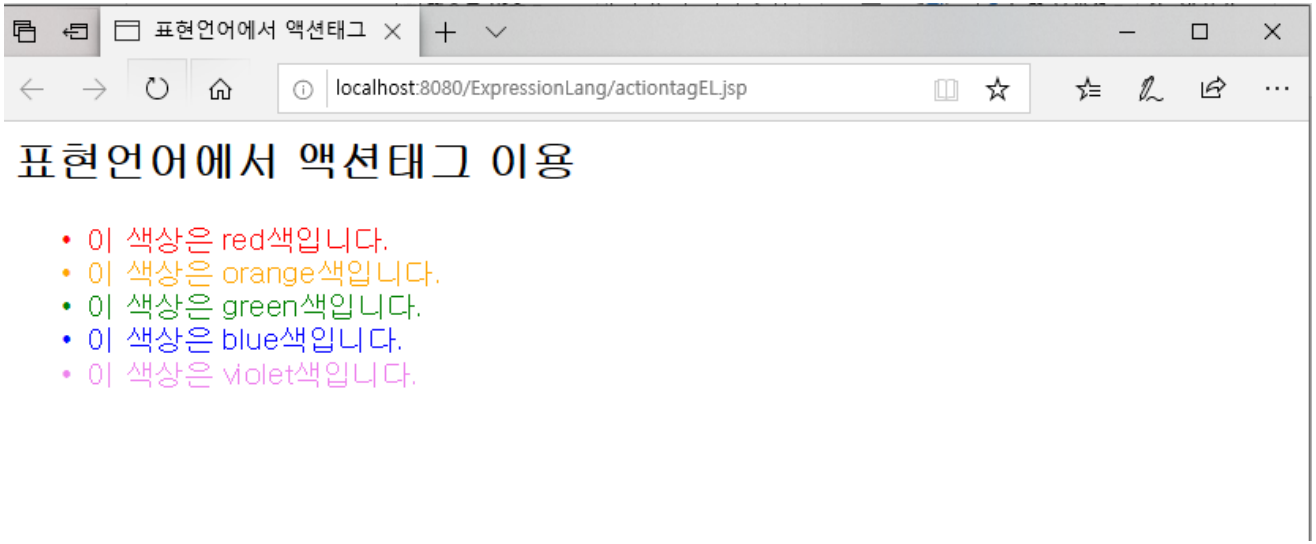
- headerObjectEL.jsp
- 404쪽 예제 12-5



액션 태그와 표현언어

ArrayList의 배열 객체 이용

- actiontagEL.jsp
- 407쪽 예제 12-7



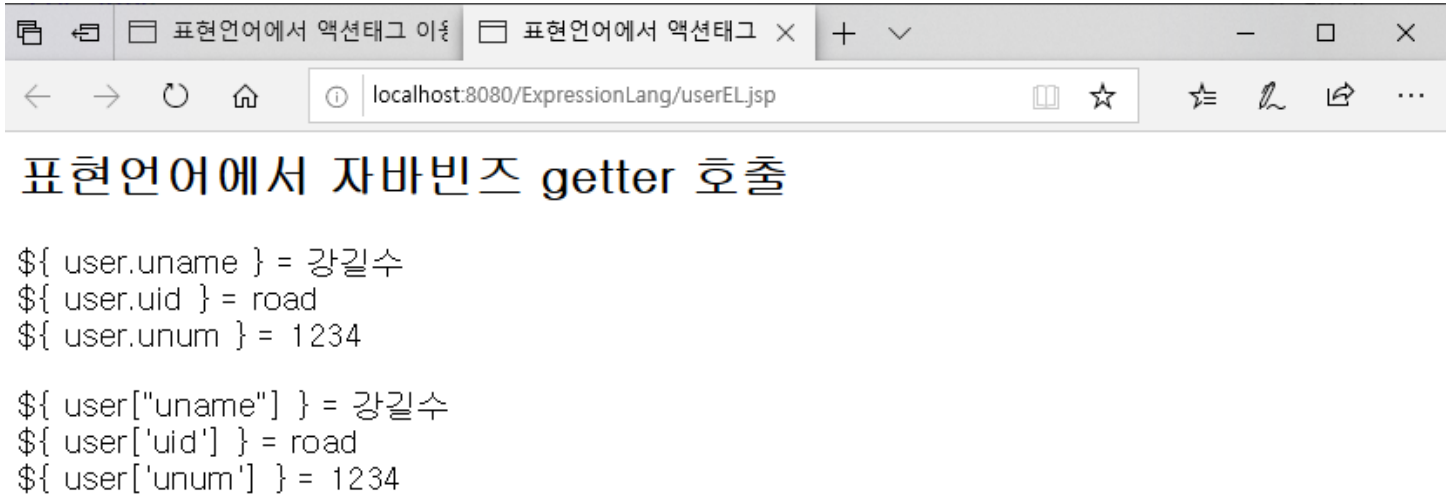
ArrayList<E> 클래스의 주요 메소드

메소드	설명
boolean add(E element)	ArrayList의 맨 뒤에 element 추가
void add(int index, E element)	인덱스 index 위치에 element 삽입
boolean addAll(Collection<? extends E> c)	컬렉션 c의 모든 요소를 ArrayList의 맨 뒤에 추가
void clear()	ArrayList의 모든 요소 삭제
boolean contains(Object o)	ArrayList가 지정된 객체를 포함하고 있으면 true 리턴
E elementAt(int index)	index 인덱스의 요소 리턴
E get(int index)	index 인덱스의 요소 리턴
int indexOf(Object o)	o와 같은 첫 번째 요소의 인덱스 리턴, 없으면 -1 리턴
boolean isEmpty()	ArrayList가 비어있으면 true 리턴
E remove(int index)	index 인덱스의 요소 삭제
boolean remove(Object o)	o와 같은 첫 번째 요소를 ArrayList에서 삭제
int size()	ArrayList가 포함하는 요소의 개수 리턴
Object[] toArray()	ArrayList의 모든 요소를 포함하는 배열 리턴

액션 태그와 표현언어

❖ 자바빈즈의 getter 호출

- User.java(409쪽 예제 12-8)
- userEL.jsp(410쪽 예제 12-9)



표현언어에서 이용할 함수 만들기

- 클래스에 정의한 메소드를 표현언어로 호출하려면
  - ${ prefixname:functioname() }
  - 먼저 접두어 prefixname으로 태그를 선언
- 표현언어에서 함수를 이용하려면 다음과 같이 3가지 작업을 수행

순서	작업	파일이 저장되는 폴더	파일이름
1	클래스 작성	[Java Resources: src]/[패키지]	ELDateFormat.java
2	TLD 파일 작성	[WebContent]/[WEB-INF]/[tld]	ELfunction.tld
3	JSP 파일 작성	[WebContent]	function.jsp

ELDateFormat.java


```
package form;

import java.text.SimpleDateFormat;
import java.util.Date;

public class ELDateFormat {
    private static SimpleDateFormat df =
                        new SimpleDateFormat("yyyy-MM-dd(E) HH:mm:ss");

    public static String toFormat(Date date) {
        return df.format(date);
    }
}
```

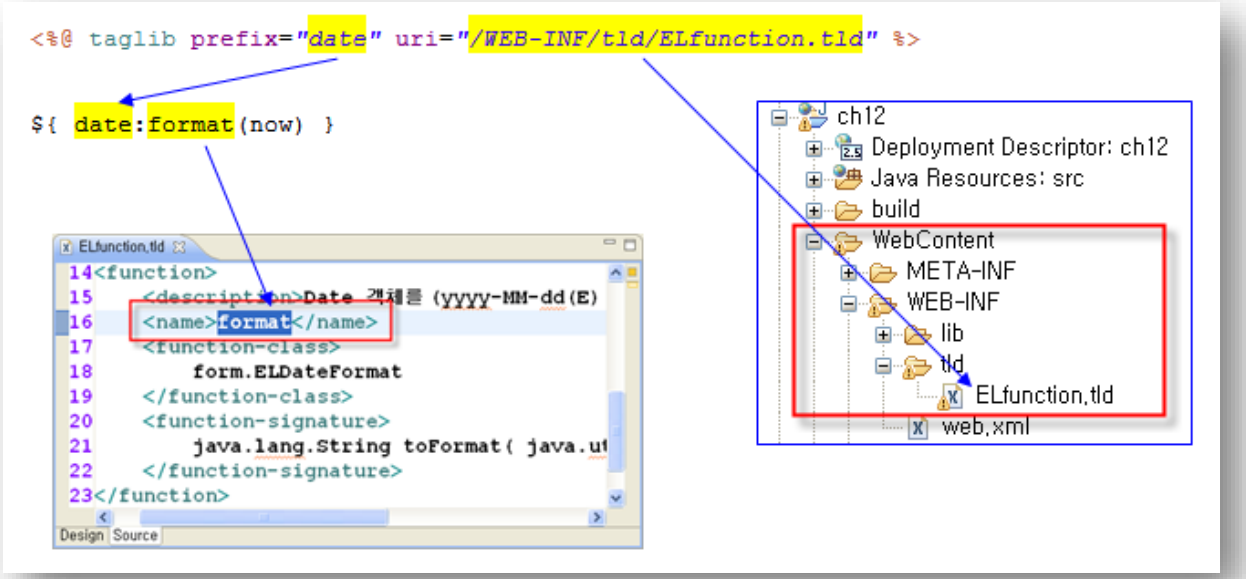
el-functions.tld

```
<?xml version="1.0" encoding="euc-kr" ?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                        web-jsptaglibrary_2_0.xsd"
    version="2.0">

    <description>EL에서 함수실행</description>
    <tlib-version>1.0</tlib-version>
    <short-name>ELfunctions</short-name>
    <uri>/ELfunctions</uri>

    <function>
        <description>Date 객체를 (yyyy-MM-dd(E) HH:mm:ss) 형태로 출력</description>
        <name>format</name>
        <function-class>
            form.ELDateFormat
        </function-class>
        <function-signature>
            java.lang.String toFormat( java.util.Date )
        </function-signature>
    </function>

</taglib>
```

function.jsp

표현언어에서 등록한 태그의 함수를 호출하려면

- 가장먼저 <taglib> 태그를 이용하여 사용할 태그 접두어와 이용할 함수가 정의되어 있는 TLD 파일을 지정



표현언어 비활성화

- 표현언어는 JSP 페이지 규약 2.0에 추가된 기능
- 만일 JSP 규약 2.0 이전 버전에서 개발된 JSP 프로그램을 JSP 규약 2.0에서 실행한다면
  - $로 시작하는 문자열을 표현언어로 인식하여 오류가 발생
  - 이러한 경우를 대비해서 JSP는 JSP 페이지에서 표현언어를 사용하지 않겠다는 표현언어 비활성화 지시를 내릴 수 있음
- 표현언어의 비활성화
  - 페이지 단위 또는 응용프로그램 단위 또는 서버 단위로 가능

표현언어 비활성화 단위	수정 내용	수정 파일
페이지 단위	페이지 지시자 속성 isELIgnored 추가	각 JSP 페이지
응용프로그램 단위	태그 <el-ignored> 추가	[WEB-INF]/web.xml
서버 단위	태그 <el-ignored> 추가	[conf]/web.xml

Thank You !

www.dongyang.ac.kr